オブジェクト指向言語・期末テスト問題用紙

（'11 年 7 月 29 日・ 10:30 〜 12:00）

解答上、その他の注意事項

I. 問題は、問 I〜VII までである。

II. 解答用紙の右上の欄に学籍番号・名前を記入すること。

III. 解答欄を間違えないよう注意すること。

IV. 解答中の文字 (特に a と d) がはっきりと区別できるよう注意すること。

V. 持ち込みは不可である。筆記用具・時計・学生証以外のものは、かばんの中などにしまうこと。

VI. テストの配点は 80 点である。合格はレポートの得点を加点して、100 点満点中 60 点以上とする。

すべての問に対する補足:
プログラムの空欄を埋める問題では、解答が長くなる可能性があるので、下の省略形（◯囲み文字）を用いても良い。例えば `this==null` と書く代わりに, Ⓣ==Ⓝと書いて良い。（必ず◯で囲むこと。）

Ⓐ `ActionListener` Ⓐ `addActionListener` Ⓒ `class` Ⓓ `actionPerformed`
Ⓔ `ActionEvent` Ⓖ `getSource` Ⓘ `implements` Ⓙ `JApplet` Ⓚ `KeyListener`
Ⓚ `addKeyListener` Ⓜ `MouseListener` Ⓜ `addMouseListener` Ⓝ `null`
Ⓟ `public` Ⓠ `equals` Ⓡ `Runnable` Ⓢ `System.out.println` Ⓣ `this`
Ⓥ `void` Ⓦ `new` Ⓧ `extends`

また、参考のために問題用紙の末尾に授業プリントの `KeyTest.java`, `UpDownButton.java`, `UpDownButton3.java`, `BubbleSort1.java`, `BubbleSort2.java`, `Point.java`, `ColorPoint.java` のソースを掲載する。

I. 次の各多肢選択問題に答えよ。解答は各問の指示する選択肢から選べ。ただし、特に指定しない限り、選ぶべき選択肢は必ずしも **1 つとは限らない**。

(i) 次のうち Java のクラスの名前として（文法的に）許されるのは、どれか?

(A). `123Daaah` (B). `Drag'n'Drop` (C). `Kagawa-U` (D). `A_1_2`

(ii) 次の Java に関する文章のうち正しいものはどれか?

(A). Java ではポインタのインクリメントなどの演算により、セキュリティ上の問題が起こる可能性がある。

(B). Java は動的束縛をデフォルトとして提供せず、その分の高速化を図っている。

(C). Java はオブジェクト指向言語であるが、C 言語や C++言語との互換性は持っていない。

(D). Web ブラウザ上でインタプリタ方式で実行する Java プログラムのことを JavaScript と呼ぶ。

(iii) Java で要素の型が `String` 型であるような、`ArrayList` 型（サイズ変更可能な配列の型）の変数 `a` を宣言したい。正しい書き方を 1 つ、以下の選択肢から選べ。

(A). `ArrayList<String> a = new ArrayList<String>();`

(B). `ArrayList[String] a = new ArrayList[String]();`

(C). `String<ArrayList> a = new String<ArrayList>();`

(D). `String[ArrayList] a = new String[ArrayList]();`

(iv) Java で 1 から 5 までの数とその 3 倍の数を次のように出力したい。

```
x が 1 のとき x*3 は 3
x が 2 のとき x*3 は 6
x が 3 のとき x*3 は 9
x が 4 のとき x*3 は 12
x が 5 のとき x*3 は 15
```

次のプログラム（の一部）:

```
int k;
for (k=1; k<=5; k++) {
    ?;
}
```

の空欄 ? にふさわしい式を下の選択肢の中から選べ。

(A). `System.out.println("xが"k"のとき x*3 は"k*3"")`

(B). `System.out.println("xが"+"k"+"のとき x*3 は"+"k*3")`

(C). `System.out.println("xが'k'のとき x*3は'k*3'")`

(D). `System.out.println("xが "+k+" のとき x*3 は"+(k*3))`

II. Scala の map は、リストのすべての要素に関数を適用し、その結果を新しいリストとして返すメソッド、filter は、与えられた真偽値関数を真にするような、リストの中のすべての要素を選択するメソッドである。

例えば、twice, odd を次のように定義された関数だとすると、

```
def twice(x: Int) = 2*x
def odd(x: Int)   = x%2 == 1
```

List(1, 2, 3).map(twice) は、List(2, 4, 6) に、List(2, 3, 4).filter(odd) は、List(3) になる。

次の式の値を以下の選択肢から選べ。

(i) `List(1, 2, 3).map((x:  Int) => x*x)`

(ii) `List(3, 4, 5).filter((x:  Int) => x*x > 10)`

選択肢（共通）

(A). List(1, 2, 3)　(B). List(1, 4, 9)　(C). List(1, 3)
(D). List(3)　(E). List(4, 5)　(F). List(9, 16, 25)
(G). 1　(H). 4　(I). 9　(J). 16　(K). 25　(L). 60

III. foo.Bar クラス（foo パッケージの Bar クラス），foo.BarTest クラス（foo パッケージの BarTest クラス），BarTest クラス（無名パッケージの BarTest クラス）をそれぞれ次のように定義する

パス: foo/Bar.java

```
package foo;

public class Bar {
  public int x;
  private int y;
  int z;

  public Bar(int a, int b, int c) {
    x = a; y = b; z = c;
  }
}
```

パス: foo/BarTest.java

```
package foo;

public class BarTest {
  public static void main(String[] args) {
    Bar bar = new Bar(1, 2, 3);
    System.out.println(bar.x);   // い
    System.out.println(bar.y);   // ろ
    System.out.println(bar.z);   // は
  }
}
```

パス: ./BarTest.java

```
import foo.Bar;

public class BarTest {
  public static void main(String[] args) {
    Bar bar = new Bar(1, 2, 3);
    System.out.println(bar.x);   // に
    System.out.println(bar.y);   // ほ
    System.out.println(bar.z);   // へ
  }
}
```

い〜へ、の行でエラーにならない行には○を、エラーになる行には×をつけよ。

IV. 次の文章は java.awt.Color クラスの brighter メソッドと darker メソッドの説明の Java™ API 仕様からの抜粋である。

> public Color **brighter**()
>
> この Color をより明るくした、新しい Color を生成します。
>
> このメソッドは、この Color の 3 つの RGB 成分のそれぞれに任意のスケーリング係数を適用することにより、この Color をより明るくした色を生成します。brighter と darker は逆の操作ですが、これら 2 つのメソッドを続けて呼び出した場合、丸め誤差のために、結果が一致しないことがあります。
>
> **戻り値:**
>
> この Color をより明るくした、 新しい Color オブジェクト
>
> public Color **darker**()
>
> この Color をより暗くした、新しい Color を生成します。
>
> このメソッドは、この Color の 3 つの RGB 成分のそれぞれに任意のスケーリング係数を適用することにより、この Color をより暗くした色を生成します。brighter と darker は逆の操作ですが、これら 2 つのメソッドを続けて呼び出した場合、丸め誤差のために、結果が一致しないことがあります。
>
> **戻り値:**
>
> この Color をより暗くした、 新しい Color オブジェクト

このメソッドを使用し、テストするプログラムを次のように作成する。また、実行例は右の図のようになる。

ファイル名: ColorTest.java

```
import java.awt.*;
import javax.swing.*;

public class ColorTest extends JApplet {
  @Override
  public void paint(Graphics g) {
    Color mycolor = new Color(96, 64, 192);
    g.setColor(          (i)          );
    g.drawString("Hello␣World!", 50, 25);
    g.setColor(mycolor);
    g.drawString("Hello␣World!", 50, 50);
    g.setColor(          (ii)         );
    g.drawString("Hello␣World!", 50, 75);
  }
}
```

このプログラムは、ある適当な色 mycolor (= new Color(96, 64, 192)) を作成し、その mycolor をより暗くした色で、座標 (50, 25) に、mycolor をより明るくした色で、座標 (50, 75) に、元の mycolor で、座標 (50, 50) に、それぞれ “Hello World!” という文字列を描画する。

上のプログラムの空欄 (i)〜(ii) を埋めよ。

V. 次のプログラム（QuoteOfTheDay クラス）は、“Prev”, “Next” という 2 つのボタンと、本日の名言を表示し、ボタンを押せば表示される名言が順繰りに（“Next” ボタンを押せば次に、“Prev” ボタンを押せば前に）変わるというアプレットである。

以下の空欄 (i) 〜 (iii) を埋めて、プログラムを完成させよ。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class QuoteOfTheDay extends JApplet (i) {
  private int index;
  static private final String[] quotes = {
    "Gravitation␣is␣not␣responsible␣for␣people␣falling␣in␣love."
       + "␣--␣Albert␣Einstein",
    "The␣best␣way␣to␣predic␣the␣future␣is␣to␣invent␣it."
       + "␣--␣Alan␣Key",
    "In␣mathematics␣you␣don't␣understand␣things.␣You␣just␣get␣used␣to␣them."
       + "␣--␣John␣von␣Neumann",
    "人苦不知足、既平朧、復望蜀。␣--␣劉秀（光武帝）"
  };
  private JButton nextButton, prevButton;

  @Override
  public void init() {
    index = 0;
    nextButton = new JButton("Next");
    prevButton = new JButton("Prev");
    (ii)
    setLayout(new FlowLayout());
    add(nextButton); add(prevButton);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString(quotes[index], 10, 70);
  }

  public void actionPerformed(ActionEvent e) {
    if ( (iii) ==nextButton) {
      index = (index+1) % quotes.length;
    } else {
      index = (index-1+quotes.length) % quotes.length;
    }
    repaint();
  }
}
```

VI. つぎに定義される BasicBlock クラスを継承して、NormalBlock クラス、さらに NormalBlock クラスを継承して、ColorBlock クラスを定義する。

ファイル: Block.java

```
1  public class BasicBlock {
2    public BasicBlock() {}  // 注： 実際にはコンストラクタは定義する必要はない。
3                            //     （デフォルトの定義と同じなので）
4    public void hit() {
5      return;
6    }
7
8    public String getColor() {
9      return "white";
10   }
11 }
```

ファイル: NormalBlock.java

```
1  public class NormalBlock [        (i)        ] {
2    private int hitPoint;
3
4    public NormalBlock(int hp) {
5      super();
6      hitPoint = hp;
7    }
8
9    public int getHitPoint() {
10     return hitPoint;
11   }
12
13   @Override
14   public void hit() {
15     if (hitPoint>0) {
16       hitPoint--;
17     }
18   }
19
20   @Override
21   public String getColor() {
22     if (hitPoint>0) {
23       return super.getColor();
24     } else {
25       return "invisible";
26     }
27   }
28 }
```

ファイル: ColorBlock.java

```
1  public class ColorBlock [         (ii)        ] {
```

```
2   private final static String[] colors = {"invisible", "red", "orange",
3                 "yellow", "green", "blue", "indigo", "violet", "white"};
4
5   public ColorBlock() {
6     super(colors.length-1);
7   }
8
9   @Override
10  public String getColor() {
11    return colors[getHitPoint()];
12  }
13 }
```

また、BlockTest クラスは、これらのクラスのテスト用の main メソッドを持つ。

ファイル: BlockTest.java

```
1  public class BlockTest {
2    public static void main(String[] args) {
3      BasicBlock[] blocks = new BasicBlock[3];
4      blocks[0]    = new BasicBlock();
5      blocks[1]    = new NormalBlock(2);
6      ColorBlock c = new ColorBlock();
7      c.hitPoint = 4;    // 注
8      blocks[2] = c;
9
10     int i, j;
11     for(i=0; i<3; i++) {
12       for(j=0; j<blocks.length; j++) {
13         System.out.print("␣" + blocks[j].getColor());
14         blocks[j].hit();
15       }
16       System.out.print("\n");
17     }
18   }
19 }
```

(i) の空白を埋めて、NormalBlock クラスの定義を完成させよ。

(ii) の空白を埋めて、ColorBlock クラスの定義を完成させよ。

(iii) BlockTest クラスの 7 行目はコンパイル時にエラーになる。
しかし、テストのため一時的に NormalBlock クラスのフィールド hitPoint の値を、この 7 行目のように他のオブジェクトのメソッドから変更できるようにしたい。
NormalBlock クラスの定義の何行めをどのように変更すれば良いか？(プログラム中の行の左の数字がクラスの中での行数を表す。)

(iv) 上記の変更を施し（つまり、BlockTest クラスの 7 行目がエラーにならないようにし)、このプログラム（BlockTest クラスの main メソッド）を実行するとき、出力はどうなるか？(空白は気にしなくて良い。)

VII. 下のプログラムは、キーボードから打ち込まれた ←（左カーソルキー）, →（右カーソルキー）の 2 つのキーに従って速度を変え移動する物体のアニメーションを描画する Java アプレットである。

ファイル: Breakout.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Breakout [ (i) ] {
  private double x = 320;
  private double dx = 0;
  private Thread aThread = null;

  @Override
  public void start() {
    if (aThread==null) {
      aThread = new Thread(this);
      aThread.start();
    }
  }

  @Override
  public void stop() {
    aThread = null;
  }

  @Override
  public void [ (ii) ]() {
    addKeyListener(this);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.fillRect((int)x, 300, 20, 5);
  }

  public void [ (iii) ]() {
    while (aThread==Thread.currentThread()) {
      x += dx;
      if (x>640) {
        x -= 640;
      } else if (x<0) {
        x += 640;
      }
      repaint();
      try {
        Thread.sleep(300);
```

```
      } catch (InterruptedException e) {}
    }
  }

  public void keyPressed(KeyEvent e) {
    int key = e.getKeyCode();
    switch (key) {
    case KeyEvent.VK_LEFT:  dx-=0.5; break;  // ← キーが押されたとき
    case KeyEvent.VK_RIGHT: dx+=0.5; break;  // → キーが押されたとき
    default:                break;
    }
  }

  public void keyReleased(KeyEvent e) {}
  public void keyTyped(KeyEvent e) {}
}
```

(i)〜(iii) の空欄を埋めてプログラムを完成させよ。

さらに、同じ動作をするプログラムを匿名（無名）クラスを KeyListener として利用して Breakout2 クラスとして実装する。(iv)〜(v) の空欄を埋めてプログラムを完成させよ。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Breakout2 [ (iv) ] {
  private double x = 320;
  private double dx = 0;
  private Thread aThread = null;

  // start, stop, paint, [ (iii) ]メソッドは Breakout と同一なので省略
  @Override
  public void [ (ii) ]() {
    addKeyListener(
        [ (v) ]
    );
  }
}
```

以下に参考のために授業配布プリントの KeyTest.java, UpDownButton.java, UpDownButton3.java, BubbleSort1.java, BubbleSort2.java, Point.java, ColorPoint.java のソースを掲載する。

KeyTest.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class KeyTest extends JApplet implements KeyListener {
  int x=50, y=20;

  @Override
  public void init() {
    addKeyListener(this);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString("HELLO␣WORLD!", x, y);
  }

  public void keyTyped(KeyEvent e) {
    int k = e.getKeyChar();
    if (k=='u') {
      y-=10;
    } else if (k=='d') {
      y+=10;
    }
    repaint();
  }
  public void keyReleased(KeyEvent e) {}
  public void keyPressed(KeyEvent e) {}
}
```

UpDownButton.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class UpDownButton extends JApplet implements ActionListener {
  int x=20;
  JButton lBtn, rBtn;

  @Override
  public void init() {
    lBtn = new JButton("Left");
    rBtn = new JButton("Right");
    lBtn.addActionListener(this);
    rBtn.addActionListener(this);
    setLayout(new FlowLayout());
    add(lBtn); add(rBtn);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString("HELLO␣WORLD!", x, 55);
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == lBtn) {         // lBtnが押された
      x-=10;
    } else if (e.getSource() == rBtn) {  // rBtnが押された
```

```
      x+=10;
    }
    repaint();
  }
}
```

UpDownButton3.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class UpDownButton3 extends JApplet {
  int x=20;

  @Override
  public void init() {
    JButton lBtn = new JButton("Left");
    JButton rBtn = new JButton("Right");
    lBtn.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        x-=10;
        repaint();
      }
    });
    rBtn.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        x+=10;
        repaint();
      }
    });
    setLayout(new FlowLayout());
    add(lBtn); add(rBtn);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(g);
    g.drawString("HELLO␣WORLD!", x, 55);
  }
}
```

BubbleSort1.java

```
import javax.swing.*;
import java.awt.*;

public class BubbleSort1 extends JApplet implements Runnable {
  int[] args = { 10, 3, 46, 7, 23, 34, 8, 12, 4, 45, 44, 52};
  Color[] cs ={ Color.RED, Color.ORANGE, Color.GREEN, Color.BLUE};
  Thread thread=null;

  @Override
  public void start() {
    if (thread == null) {
      thread = new Thread(this);
      thread.start();
    }
  }

  @Override
  public void stop() {
    thread = null;
  }

  @Override
```

```
  public void paint(Graphics g) {
    int i;

    super.paint(g);
    for(i=0; i<args.length; i++) {
      g.setColor(cs[args[i]%cs.length]);
      g.fillRect(0, i*10, args[i]*5, 10);
    }
  }

  public void run() {
    int i, j;
    Thread thisThread = Thread.currentThread();

    for (i=0; i<args.length-1; i++) {
      for (j=args.length-1; thread == thisThread && j>i; j--) {
        if (args[j-1]>args[j]) { // スワップする。
          int tmp=args[j-1]; args[j-1]=args[j]; args[j]=tmp;
        }
        repaint();
        try { // repaint の後でしばらく止まる
          Thread.sleep(500);
        } catch (InterruptedException e) {}
      }
    }
  }
}
```

BubbleSort2.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class BubbleSort2 extends JApplet implements Runnable, ActionListener {
  int[] args = { 10, 3, 46, 7, 23, 34, 8, 12, 4, 45, 44, 52};
  Color[] cs ={Color.RED, Color.ORANGE, Color.GREEN, Color.BLUE};
  Thread thread=null;
  private boolean threadSuspended=true;

  @Override
  public void init() {
    JButton step = new JButton("Step");
    step.addActionListener(this);
    setLayout(new FlowLayout());
    add(step);
  }

  // start, stop, paint メソッドは BubbleSort1.java と同一なので省略する。

  public synchronized void actionPerformed(ActionEvent e) {
    threadSuspended=false;
    notify();
  }

  public void run() {
    int i, j;
    for (i=0; i<args.length-1; i++) {
      for (j=args.length-1; j>i; j--) {
        if (args[j-1]>args[j]) { // スワップする。
          int tmp=args[j-1]; args[j-1]=args[j]; args[j]=tmp;
        }
        repaint();
        try {  // repaint の後で止まる
          synchronized(this) {
```

```
            while (threadSuspended) {
              wait();
            }
            threadSuspended=true;
          }
        } catch (InterruptedException e) {}
      }
    }
    thread=null;
  }
}
```

Point.java

```
public class Point {
  public int x, y;

  public void move(int dx, int dy) {
    x += dx;  y += dy;
  }

  public void print() {
    System.out.printf("(%d,␣%d)", x, y);
  }

  public void moveAndPrint(int dx, int dy) {
    print(); move(dx, dy); print();
  }

  public Point(int x0, int y0) {
    x = x0; y = y0;
  }
}
```

ColorPoint.java

```
public class ColorPoint extends Point {
  private String[] cs = {"black", "red", "green", "yellow",
                         "blue", "magenta", "cyan", "white"};
  private Color color;     // 0-黒 1-赤 2-緑 3-黄 4-青 5-紫 6-水 7-白

  @Override
  public void print() {
    System.out.printf("<font␣color='%s'>", getColor());  // 色の指定
    super.print();
    System.out.print("</font>");                          // 色を戻す
  }

  public void setColor(String c) {
    int i;
    for (i=0; i<cs.length; i++) {
      if (c.equals(cs[i])) {
        color = c; return;
      }
    } // 対応する色がなかったら何もしない。
  }

  public ColorPoint(int x, int y, String c) {
    super(x, y);
    setColor(c);
  }

  public String getColor() {
    return color;
  }
}
```

オブジェクト指向言語・期末テスト解答用紙（'11 年 7 月 29 日）

学籍番号		氏名	

I. （3 × 4）

(i).		(ii).		(iii).		(iv).	

II. （3 × 2）

(i).		(ii).	

III. （6 つ正解で 6 点, 5 つ正解で 4 点, 4 つ正解で 2 点）

い	ろ	は	に	ほ	へ

IV. （4 × 2）

(i).	
(ii).	

V. （4 × 3）

(i).	
(ii).	
(iii).	

VI. （4 × 4）

(i).	
(ii).	
(iii).	
(iv).	

VII. （4×5）

(i).	
(ii).	
(iii).	
(iv).	
(v).	

授業・テストの感想

……

……

……

……

……

……

……

……

……

……

……